AMENDMENTS TO THE CLAIMS:

This listing of claims will replace all prior versions, and listings, of claims in the application:

Listing of the Claims:

1. (Currently Amended) A system, comprising:

a first processor;

a second processor coupled to the first processor;

~~an~~ single operating system configured to execute exclusively on the first processor; and

a middle layer software configured to execute on the first processor and configured to distribute tasks to run on either or both processors.

2. (Original) The system of claim 1 wherein the middle layer software comprises a Java virtual machine.

3. (Previously Presented) The system of claim 1 further comprising a synchronization unit coupled to the first and second processors, said synchronization unit configured to synchronize the execution of the first and second processors.

4. (Previously Presented) The system of claim 3 wherein the synchronization unit is configured to cause the first processor to transition to a wait mode while the second processor executes a task.

5. (Previously Presented) The system of claim 4 wherein the first processor is configured to transition from the wait mode to a fully operational mode by a signal asserted by the either the first or second processor to the synchronization unit.

6. (Previously Presented) The system of claim 1 further comprising a shared TLB configured to contain a plurality of entries in which virtual-to-physical address translations are stored, each entry also comprising a task ID field in which a task ID associated with the corresponding translation and with a task running on the first or second processor is stored.

7. (Previously Presented) The system of claim 6 wherein the operating system is configured to selectively flush at least one of the entries in the shared TLB based on task ID.

8. (Previously Presented) The system of claim 6 wherein the middle layer software is configured to selectively flush at least one of the entries in the shared TLB based on task ID.

9. (Previously Presented) The system of claim 8 wherein the middle layer software comprises a Java virtual machine.

10. (Previously Presented) The system of claim 6 wherein at least one of the shared TLB entries are invalidated, and those entries that are invalidated have task IDs that are associated with tasks that are running or have run on only one of the first or second processors.

11. (Previously Presented) The system of claim 1 wherein the second processor has a programmable context and is configured to autonomously switch its own context without support from the operating system executing on the first processor.

12. (Previously Presented) The system of claim 1 wherein the second processor includes a programmable task ID register which is configured to contain a value indicative of the task currently running on the second processor that is written by the middle layer software running on the first processor.

13. (Currently Amended) A method usable in a multi-processor system, comprising:
executing ~~an~~ single operating system on only one of a plurality of processors; and
distributing tasks to each of the plurality of processors by middle layer software running on the processor on which the operating system executes.

14. (Previously Presented) The method of claim 13 wherein distributing tasks comprises distributing tasks by a Java virtual machine.

15. (Previously Presented) The method of claim 13 further comprising causing the processor on which the operating system executes to transition to a wait mode while another processor executes tasks and subsequently transitioning the processor in the wait mode to an active mode as a result of a signal being asserted by any of the plurality of processors.

16. (Previously Presented) The method of claim 13 wherein each task has a unique task identifier value and the method further comprises writing virtual-to-physical address translations and task identifier values associated with the task to which the translations pertain into a translation lookaside buffer that is shared between the plurality of processors.

17. (Previously Presented) The method of claim 16 further selecting task identifier values and invalidating entries in the translation lookaside buffer that contain the selected task identifier values and not invalidating other entries in the translation lookaside buffer.

18. (Previously Presented) The method of claim 13 wherein, in a processor having a context and that does not execute the operating system, autonomously switching said context without support from the operating system.

19. (Previously Presented) The method of claim 13 further comprising writing a task ID register by the processor executing the operating system, the task ID register contained in another processor.

20. (Currently Amended) A computer-readable medium storing a program that, when executed by a multi-processor system, causes only one of a plurality of processors to:

execute ~~an~~ single operating system; and

distribute tasks to each of the plurality of processors by middle layer software that runs on the processor.

21. (Previously Presented) The computer-readable medium of claim 20 wherein when the processor distributes, the program causes the processor to distribute tasks by a Java virtual machine.

22. (Previously Presented) The computer-readable medium of claim 20 wherein when the processor executes, the program causes the processor to transition to a wait mode while another processor executes tasks and subsequently transitions the processor in the wait mode to an active mode as a result of a signal being asserted by any of the plurality of processors.

23. (Previously Presented) The computer-readable medium of claim 20 wherein each task has a unique task identifier value; and when the processor writes, the program causes the processor to write virtual-to-physical address translations and task identifier values associated with the task to which the translations pertain into a translation lookaside buffer that is shared between the plurality of processors.

24. (Previously Presented) The computer-readable medium of claim 23 wherein when the processor selects, the program causes the processor to select task identifier values and invalidate entries in the translation lookaside buffer that contain the selected task identifier values and not invalidate other entries in the translation lookaside buffer.

25. (Previously Presented) The computer-readable medium of claim 20 wherein in a processor having a context and that does not execute the operating system, the program causes the processor to autonomously switch said context without support from the operating system.

26. (Previously Presented) The computer-readable medium of claim 20 further comprises when the processor writes, the program causes the processor to write a task ID register, the task ID register contained in another processor.